Toward a Modular Cassandra

Derek Chen-Becker

ApacheCon North America, 2022

About Me

- Senior Engineer at AWS
- Lead Maintainer, SigV4 Auth Plugin
- Advocate and enthusiast
  - Involved in OSS 15 years
  - Working with Cassandra 7+ years
  - Involved more in the last 3

Agenda

- Why Modularity?
- Existing Work
- Storage API Proposal
- More Modularity
- Call to Action

Modularity

Why Modularity?

- Give end users flexibility and choice
  - Tune to workload
  - Additional features
- Experimentation and improvement
  - Ideas don't have to be holistic
  - Easier to focus on a specific area

Reduce Complexity

- Make it easier for devs to understand the system
- Make it easier to audit
- Lower the threshold for meaningful contributions
- Currently difficult (for me) to figure out boundaries

Trends In Modularization

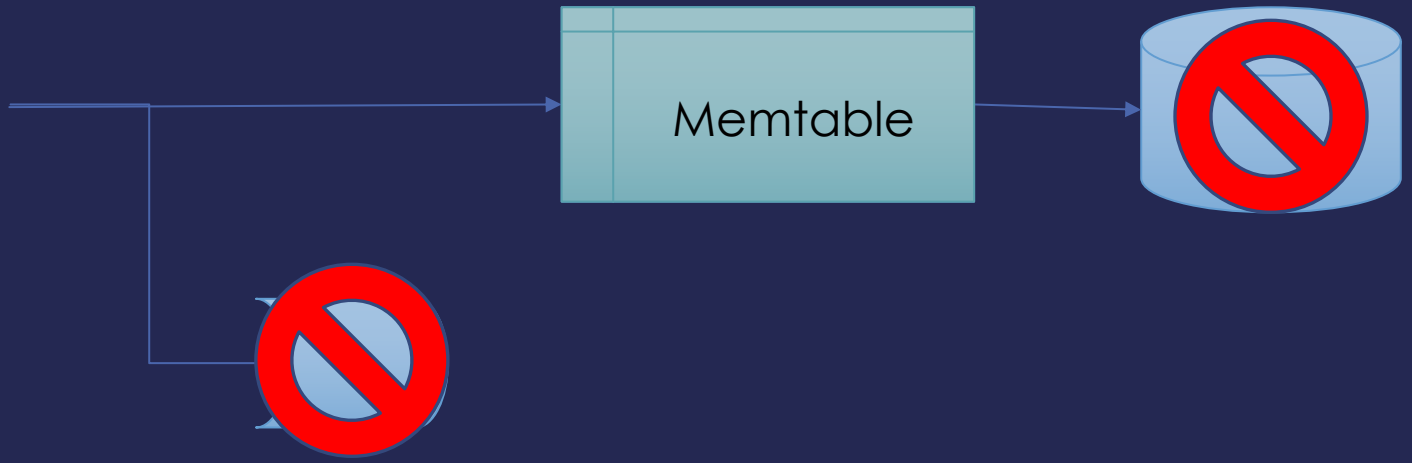
CASSANDRA-13475

- 5+ years old, not much progress lately
- Pluggable Storage Engine API
- RocksDB Implementation

CEP-11 : Pluggable Memtable

- Alternate off-heap memtable to reduce GC impact
- Includes both a new interface for Memtables, as well as a new implementation
- Indirectly impacts commit log and durable store

CEP-11 : Pluggable Memtable



CEP-18 : Modularization

- Targets 4 areas: Schema, Topology, Compaction, and Failure Detection
- Withdrawn in favor of per-functionality tickets
- Schema pluggability and failure detection merged

CEP-21 : Cluster Metadata Service

- Main motivation is to address existing failure modes
- Proposes linearization of cluster metadata (schema, topology, etc)
- Does not propose externalizing metadata, but also does not preclude

Codebase Challenges

It's Not You, It's Me

- Challenges specific to modularization
- Partly, coming up to speed on a mature codebase
- *Potentially* an issue for other new contributors

"Statics all the way down..."

Not just Schema

```
~/c/cassandra ❯❯❯ rg -Uc 'static.*\sinstance\s+=' | wc -l
156
```

- DatabaseDescriptor is essentially one giant global variable
- Good news
  - Only a small number are related to pluggable behavior
  - Gradual change is OK

Diffusion of Responsibility

- Different aspects of processing queries are commingled
- Figuring out what a given method/class does can be complicated
- Impedes testing and reasoning about changes

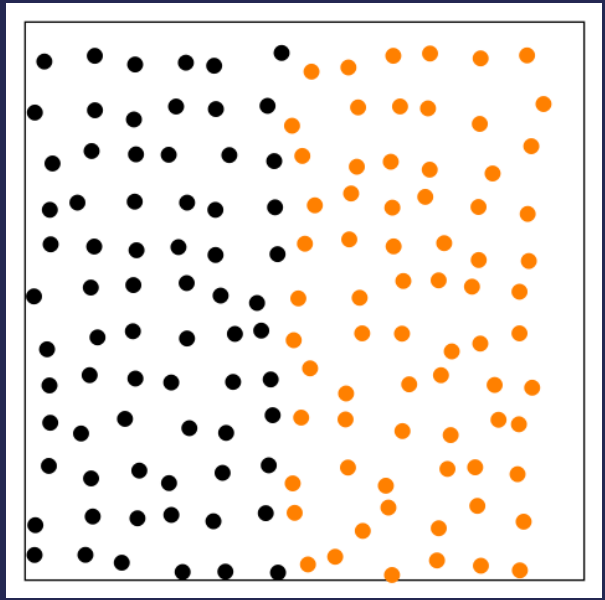


Image from https://commons.wikimedia.org/wiki/File:Diffusion_microscopic_picture.gif, Jacopo Bertolotti

Interfaces, kind of

- `SchemaProvider` has 9 methods
- `Schema` has 40 public methods, and only implements `SchemaProvider`

Why does this matter?

- Lack of clear separation of concerns reduces comprehensibility
- There are definitely valid reasons for statics
- Not uncommon in long-lived projects as they evolve
- Difficult to experiment
  - CEP-18 Schema: 3,164 additions and 1,913 deletions
  - CASSANDRA-13475 (pluggable storage API): open since 2017

Modularizing Storage

Why?

- Better match user requirements
- Allows for easier experimentation
- Precedence in other DBs
  - MySQL
  - PostgreSQL
  - MongoDB

Step 1: Questioning Identity

- Fundamental
  - CQL/Schema
  - Transactions
- Important, but tradeable
  - User-selectable consistency level
  - CDC
- Important, but
  - Performance characteristics

Step 2: Draw a Line

- How much do we want behind the abstraction?
- Do we want a single API, or can we compose things?
- Can we have multiple layers of modularity?

Read Path Overview

- Parse and validate query
- Transform restrictions and key bounds
- Dispatch to `StorageProxy` to get `PartitionIterator`
- Translate `PartitionIterator` into `ResultSet`

Write Path Overview

- Parse and validate query
- Transform conditions
- Check disk space (!)
- Dispatch to `StorageProxy`

StorageProxy

- It's in the name…
- Much larger than the interface suggests
- Abstracts a lot of functionality
  - Commit log
  - Replication
  - Hinting

Challenges

- Need to get agreement on essential/pluggable aspects
- Breaking work down into incremental phases
- Figure out how to involve new API

What About CDC?

- Current implementation is minimal functionality
  - Hard link commit logs
  - Build index for commit logs
  - To the consumer: caveat emptor!
- Can we design an interface?
- Should this be tied to the storage engine?

Authentication and Authorization

- AuthN and AuthZ already pluggable (plaintext, Kerberos, LDAP, SigV4)
- Room for improvement in terms of granularity and model: consider a change from listing permissions to authorizing per call
- Pluggable AuthN for inter-node?

What Next?

Call to Action

- Audit existing interactions (in progress) with storage
- Before reaching CEP, need lots of discussion
- Reach out if interested!
- apache@chen-becker.org

Questions?